Incorporating Domain-Specific Knowledge into Evolutionary Algorithms

Jinxing Xie and Wenxun Xing
Dept. of Applied Mathematics, Tsinghua University, Beijing 100084, P. R. China
Email: jxie@math.tsinghua.edu.cn, wxing@math.tsinghua.edu.cn

Abstract. Recently, the theoretical understanding of search algorithms, especially the so-called "No Free Lunch" Theorem, shows certain fundamental limitations on universal search algorithms. One of the practical implications of these understandings is exploiting domain-specific knowledge and incorporating them into search algorithms. This paper emphasises the fundamental importance of the ways in which such knowledge may be systematically exploited and incorporated into evolutionary algorithms, and suggests that domain-specific representation scheme and domain-specific local search be two basic starting points to combine problem-related knowledge and the general structure of evolutionary algorithms. A heuristic genetic algorithm for multilevel capacitated lot-sizing problem is used as an example to demonstrate these principles.

Key Words. Evolutionary Algorithms, No Free Lunch Theorem, Domain-Specific Knowledge, Capacitated Lot-Sizing Problems

1 Introduction

Stochastic search algorithms borrowing ideas from physical and biological metaphors grow rapidly during last two decades. The methods of simulated annealing (Kirkpatrick et al., 1983) and tabu search (Glover, 1986) are mainly stimulated by physical analogues. Other randomised methods, now generally entitled as evolutionary algorithms, from the seminal work on genetic algorithms and classifier systems (Holland, 1975), evolution strategies (Rechenberg, 1973, 1984; B黺k & Schwefel, 1993) and the evolutionary programming (Fogel et al., 1966), to more recent developments of the genetic programming (Koza, 1991), memetic algorithm (Pablo & Norman, 1992; Radcliffe & Surry, 1994), etc., are stimulated by natural selection and evolutionary theory. Evolutionary algorithms simply simulate the biological evolution on computer and operate on a population of individuals (candidate solutions) which manipulated by genetic operators. The linkages of the simulation to optimisation objective are built via a biased sampling methods, i.e. a fitness-based selection process, where the fitness value of an individual depends on its quality with respect to the optimisation objective. One of the advantages of evolutionary algorithms over classical optimisation or search algorithms is their global optimisation ability for multi-modal problems regardless of the properties of the objective function, such as convexity and differentiability. Other advantages include intrinsic parallelism, convergence with probability one, simplicity in implementation on computer and so on (Goldberg, 1989).

Due to their theoretically powerful ability to find global optimisation solutions for most of the optimisation problems in a variety of fields, and their simplicity to be understood by everyone and implemented on computers, evolutionary algorithms are regarded as most preferable and universal optimisation methods by many practitioners. The philosophy "trying an evolutionary algorithm if you cannot find other methods" is a bit popular in current research and practice. Even more, some practitioners accept such philosophy as "using an evolutionary algorithm even if you can find an efficient classical algorithm" or "it is unnecessary to consider domain-specific knowledge of interest when using evolutionary algorithms". These popular trends lead to some misuse of evolutionary algorithms and make some practitioners confused with how to choose a correct search algorithm for their specific problem. Despite occasional warnings from various researchers a great deal of research seems oblivious to the fact that in certain situation there is no scope for distinguishing any of these biased sampling methods from enumeration according to so瑺alled "No Free Lunch" Theorem(NFLT) (Wolpert & Macready, 1995; Radcliffe & Surry, 1995). NFLT broadly shows that all search algorithms perform equally well if a sufficiently inclusive class of problems is considered, explicitly reveals fundamental limitations on search algorithms, and particularly highlights the necessity of theory exploiting and incorporating domain-specific knowledge (Radcliffe & Surry, 1995).

This paper emphasises the fundamental importance of the ways in which domain-specific knowledge may be systematically exploited and incorporated into evolutionary algorithms, and suggests that the domain-specific representation scheme and the domain-specific local search be two basic starting points to combine problem-related knowledge and the general structure of evolutionary algorithms. A heuristic genetic algorithm for capacitated lot-sizing problem is used as an example to demonstrate these principles.

2 No Free Lunch Theorem and Its Implications

2.1 *Search Problem and Search Algorithm*

NFLT is of the performance of a search algorithm for a search problem. A search problem (or called optimisation problem) *D* is consisted of descriptions of a solution space *S* (the set of candidate objects over which the search is to be conducted), an objective function *f* (which is a mapping from *S* to the space of real objective function values, *R*) and the goal of maximisation (or minimisation) of *f*. It is such description that includes all the domain-specific knowledge and an instance of *D* can be denoted by (*S, f*). A search algorithm (either deterministic or stochastic) for the problem domain *D* is an iterative strategy to produce a final solution from *S* given any instance (*S, f*) in *D*. As the key component of a search algorithm, the iterative strategy will be defined by a function which, given a sequence of points from *S*, and the corresponding sequence of objective function values for these points, (either deterministically or stochastically) generates a new point from *S*.

As population-based stochastic search algorithms, evolutionary algorithms distinguish from other search algorithms by operating on chromosome space *I* (or representation space of *S*) instead of *S* itself. The representation space *I* of *S* will be taken to be a set of size at least equal to *S*, together with a surjective function *g* (the growth function) that maps *I* onto *S*. Objective function values will also be associated with points in the representation space *I* through composition of *f* with *g*.

2.2 No Free Lunch Theorems

NFLT was firstly presented by Wolpert and Macready (1995), and Radcliffe and Surry (1995) generalised it to a more accessible and general form. Strictly, this theorem shows that over the ensemble of all representations of one space with another, all search algorithms (in a rather broad class) perform identically on any reasonable measure of performance. One corollary of the theorem is that no non璪epeating search algorithm (algorithm does not revisit points) can outperform enumeration over the ensemble of all problems.

A first glance of this theorem will be not optimism for the fundamental limitation addressed by the theorem. All of the previous continuous endeavour aiming to find an outperforming algorithm for all the optimisation problems is questionable and should be carefully reassessed. But there are many possible improvements that are not strongly limited by the theorem, for example, revisiting points previously visited is popular in practical search algorithms. More important, the theorem states overall performance of an algorithm for all problems and assumes the algorithm make no use of domain-specific knowledge. Relaxing these restrictions may dramatically improve the performance of a search algorithm. So it is not a really bad news.

2.3 Implications of No Free Lunch Theorem

Many useful implications can be drawn from NFLT. For example, more attention should be paid on the scope and reasonability of the algorithm evaluation. The evaluation of an algorithm and the comparison of some related algorithms, either analytically or numerically, are daily work for algorithmic researchers. According to NFLT, more reasonable comparisons could be conducted between different search algorithms when the problem domain is clearly specified. For a specified search algorithm, distinguishing two problem domains with one performs well and the other performs badly is also reasonable and valuable. These observations also provide a motivation for considering ways in which both good representations and good algorithms can be designed and recognised for particular problem classes (Radcliffe and Surry, 1995).

One of the most significant implications of NFLT is that algorithms should be matched to the search problem at hand. Although a formal representation and many formal genetic algorithms can be defined independent of problem-domain (Radcliffe and Surry, 1994), this formal evolutionary algorithm without any domain-knowledge can not hope to perform well. For a formal evolutionary algorithm, "if no domain瑗pecific knowledge is used in selecting an appropriate representation, the algorithm will have no opportunity to exceed the performance of an enumerative search" (Radcliffe and Surry, 1995).

Generally speaking, a more satisfied solution can be gained with more deep domain-knowledge incorporated. But the domain-knowledge of a specific optimisation problem is usually too vast to be considered completely with a single algorithm. Even worse, there may be some domain-knowledge which is difficult to be considered explicitly when design an evolutionary algorithm. These difficulties reveal that, when designing a specific evolutionary algorithm for a specific problem domain, the critical problem remaining is how to exploit the domain-specific knowledge and how to incorporate them into the general logic and structure of evolutionary algorithms. Next section will discuss in more details about the possible starting points.

3 Incorporation of Domain-Specific Knowledge

3.1 General Evolutionary Algorithm

The general logic and structure of evolutionary algorithms can be expressed mathematically as follows (B點k, 1996, pp.63-64):

"An evolutionary algorithm is defined as an 8-tuple EA = (*I*, Φ, Ω, ψ, *s*, ι, μ, λ) where *I* = $A_x \times A_s$ is the space of individuals, and A_x, A_s denotes arbitrary sets. Φ: *I* → *R* denotes a fitness function assigning real values to individuals. Ω = $\{\omega_{\Theta_1}, \cdots, \omega_{\Theta_z} \mid \omega_{\Theta_i} : I^{\lambda} \to I^{\lambda}\} \cup \{\omega_{\Theta_0} : I^{\mu} \to I^{\lambda}\}$ is a set of probabilistic genetic operators ω_{Θ_i} , each of which is controlled by specific parameters summarised in the sets $\Theta_i \in R$. $s_{\Theta_s} : (I^{\lambda} \cup I^{\mu+\lambda}) \to I^{\mu}$ denotes the selection operator, which may change the number of individuals from λ or $\lambda + \mu$ to μ, where $\lambda, \mu \in N$ and $\lambda = \mu$ is permitted. An additional set Θ_s of parameters may be used by the selection operator. μ is the number of parent individuals, while λ denotes the number of offspring individuals. Finally, ι: {true, false} is a termination criterion for the EA, and the generation transition function $\Psi : I^{\mu} \to I^{\mu}$ describes the complete process of transforming a population P into a subsequent one by applying genetic operators and selection: $\Psi = s \circ \omega_{\Theta_{i_1}} \circ \cdots \circ \omega_{\Theta_{i_j}} \circ \omega_{\Theta_i}$, $\Psi(P) = s_{\Theta_s}(Q \cup \omega_{\Theta_{i_1}}(\cdots(\omega_{\Theta_{i_j}}(\omega_{\Theta_i}(P)))\cdots))$. Here $\{i_1, \cdots, i_j\} \subset \{1, \cdots, z\}$ and $Q \in \{\varnothing, P\}$."

Furthermore, it is possible to assume the genetic operators Ω and selection operator *s* be representation-independent in evolutionary algorithms (Surry & Radcliffe, 1996). As just mentioned in above section, a problem domain *D* consists of a set of problem instances and each instance defines a search space *S* of candidate solutions and an objective function *f: S* → *R*. The comparison of these ingredients of the problem domain with the general structure of EA makes clear that EA operates on representation space *I* instead of original domain solution space *S* and EA processes individuals based on fitness function Φ instead of original domain objective function *f*. This observation leads to some principal understandings on how to incorporate domain knowledge into evolutionary algorithms. Probably the most directly method is to carefully design the representation space *I* and the associated growth (or decoding) function *g: I* → *S* (fitness function Φ = $f \circ g$ *: I* → *R*) by making use of the domain-specific knowledge.

3.2 Domain-Specific Representations

The most widely used technique to combine the domain-knowledge into evolutionary algorithms is to design an encoding scheme for the solution space of the problem. Even though an universal representation is possible for all problems, it could not be an efficient one. A representation good for one problem domain may be extremely inconvenient and inefficient for another. For each specific problem, there exist numerous encoding schemes in which some of them are natural and the others may be tremendously unnatural. Even more, some of them use little knowledge of the problem domain and the others may have grasped most of the important characteristics of the problem. Randomly accepting a representation for the problem may result in inefficiency.

The travelling salesman problem (TSP) is a very good example. TSP can be presented as a 0-1 integer programming over a graph (Laporte, 1992). The decision variable $x_{ij} = 1$ means that the salesman goes through the path from city *i* to city *j*, otherwise the salesman does not go through the path.. If the 0-1 encoding is the universal representation, then it is natural to encode a solution (tour) with *n* cities into a 0-1 vector with $n \times n$ dimensions. Moreover if every two cities have a path, there are totally $2^{n \times n}$ 0-1 vectors in the binary representation space. Obviously many vectors are infeasible. How many vectors are needed to represent all feasible solutions of the TSP? Using an obvious domain-specific knowledge, a feasible solution (tour) can be represented as a permutation of (*1,2,...,n*), an order of the salesman's travelling. There are at most *n*! individuals in permutation representation space! Therefore we save $2^{n \times n}$/*n*! times of the size of representation space by using the domain-specific knowledge.

It is the user's responsibility to deeply exploit the domain knowledge, carefully analyse the advantages of different candidate encoding schemes and choose an appropriate one from them. Although this idea seems rather direct and easy to understand, it's usually neglected by some practitioners.

3.3 Domain-Specific Local Search

Another technique to combine the domain-knowledge into evolutionary algorithms is to include local search into the algorithmic logic. Based on the idea of searching over a subspace of local-optima, hill瑎limbers derived from domain-related knowledge can be applied to each individual of the population. At least two different types of incorporation of the local search into evolutionary algorithms can be differentiated. One is the so-called memetic algorithm where the local search is applied to each individual of the population and the resulting improvement replaces the original one or the worst one in the population. Such kind of local search can be achieved by a modified formal mutation operator within the framework of the formal memetic algorithm (Radcliffe & Surry, 1994). The other is the so-called

Baldwin effect where the local search is applied to each individual of the population, but instead of replacing the original individual with the resulting improvement, the fitness value of the resulting improvement is transferred to the original individual (Whitley et al., 1994). Such kind of local search can be achieved by incorporating the local search in the fitness evaluation for an individual. These analyses show that the technique of the domain-specific local search is actually a special case of the technique of the domain-specific representation.

4 Case Study: Capacitated Lot-Sizing Problem

4.1 Formulation of Capacitated Lot-Sizing Problem

In this section, we will demonstrate the principles in last section by using a heuristic genetic algorithm to multilevel capacitated lot-sizing problem (CLSP). CLSP can be mathematically formulated as the following mixed integer programming (interpretations to the variables and constraints are omitted here for simplicity, see for example Maes et al., 1991; Xie, 1995):

$$(CLSP) \qquad Min\ COST(Y,X,I)=\sum_{i=1}^{N}\sum_{t=1}^{T}\{s_{it}Y_{it}+c_{it}X_{it}+h_{it}I_{it}\} \tag{1}$$

$$s.t. \qquad I_{i,t-1}+X_{it}-I_{it}=d_{it}+\sum_{j\in S(i)}r_{ij}X_{jt}, \qquad i=1,\cdots,N,t=1,\cdots,T, \tag{2}$$

$$\sum_{i=1}^{N}(a_{kit}X_{it}+A_{kit}Y_{it})\le C_{kt}, \qquad k=1,\cdots,K,t=1,\cdots,T, \tag{3}$$

$$Y_{it}=\begin{cases}0, & if \quad X_{it}=0,\\ 1, & if \quad X_{it}>0,\end{cases} \qquad i=1,\cdots,N,t=1,\cdots,T, \tag{4}$$

$$I_{it},X_{it}\ge 0, \qquad i=1,\cdots,N,t=1,\cdots,T, \tag{5}$$

$$Y_{it}\in\{0,1\}, \qquad i=1,\cdots,N,t=1,\cdots,T. \tag{6}$$

The decision variables in CLSP are X_{it}, Y_{it} and I_{it}, among which Y_{it} is a 0-1 integer variable and X_{it}, I_{it} are positive real variables. For the complicated structure of the solution space and the constraints, it is inconvenient to encoding all these variables and use penalty to constraints' conflicts. It is easily to be seen that variables X_{it}, Y_{it} and I_{it} are not independent and we hope X_{it}, I_{it} can be heuristically worked out from variables Y_{it} by making use of the problem-specific knowledge of the lot-sizing problems. This leads to consider the setup pattern (variables Y_{it}) as only decision variables, then the most important thing during designing a genetic algorithm to CLSP is to decode the values of X_{it} and I_{it} from Y_{it} and the known parameters of the lot-sizing problem.

According to Afentakis and Gavish (1986), there exists following property (usually named "Zero-Switch" property) for uncapacitated lot-sizing problem, a simplified version of CLSP: There is an optimal solution to uncapacitated lot-sizing problem in which $x_{it}I_{i,t-1}=0$. Making use of this important domain-specific knowledge of the uncapacitated lot-sizing problems, we can heuristically build up the relationships between real number variables and Y_{it} and the known parameters of the problem. Given a production pattern (setups sequence) $\{Y_{it},\ i=1,\cdots,N,\ t=1,\cdots,T\}$, the production lot-sizes can be determined according to "Zero-Switch" Property as following :

(i) If $Y_{i\tau}=0$, then $X_{i\tau}=0$;

(ii) If $Y_{i\tau_1}=1,\ Y_{i\tau_2}=1\ (\tau_1<\tau_2\le T)$ and $Y_{i\tau}=0$ for all $\tau_1<\tau<\tau_2$, then

$$X_{i\tau_1}=\sum_{\tau=\tau_1}^{\tau_2-1}(d_{i\tau}+\sum_{j\in S(i)}r_{ij}X_{j\tau}). \tag{7}$$

That is to say, whenever a setup is introduced for an item we produce enough quantity for this item to satisfy the demands of integer number of periods between two successive setups. When no setup is activated in period 1 for a product (item), this processing method may lead to backlogging in the first few periods for this product and its successive parent products. In order to assure feasibility, the penalty method is used to deal with the constraints that no backlogging is allowed. That is to say, we can change the objective function to include penalties for backlogging. Before computing the fitness value of each individual, the objective function values $COSTU(Y,X,I)$ is calculated according to the following objective function:

$$COSTU(Y,X,I)=\sum_{i=1}^{N}\sum_{t=1}^{T}\{s_{it}Y_{it}+c_{it}X_{it}+h_{it}I_{it}\}+\beta\sum_{i=1}^{N}\sum_{t=1}^{T}[\max\{0,-I_{it}\}]^2 \tag{8}$$

where β is the penalty coefficient (a large enough positive number).

For capacitated lot-sizing problems, the optimal solution may not satisfy "Zero-Switch" property. In fact, the real number variables X_{it}, I_{it} do not only dependent on the production pattern (setups sequence) Y_{it} but also dependent on the available resources capacity C_{kt}. In this situation, one can firstly determine X_{it}, I_{it} from Y_{it} without considering the resource constraints and take this lot-size schedule as an initial plan which will be further modified by considering the resources constraints. For example, one can modify this initial lot-size plan by "shifting" techniques which has been widely used in the heuristics for the lot-sizing problems (Clark & Armentano,1995; Roll & Karni, 1991).

The shifting procedure checks capacity feasibility of the initial lot-size schedule from the last period backwards to period 1. In each period *t*, a capacity tightness index, ρ_{kt}, which is defined to be the ratio of the total capacity for a resource needed to produce all the scheduled items in this period to the total available capacity for this resource at this period, is calculated for each resource *k:*

$$CN_{kt} = \sum_{i=1}^{N} (a_{kit} * X_{it} + A_{kit} * Y_{it}), \rho_{kt} = CN_{kt} / C_{kt}, \qquad (k = 1,2,...,K). \tag{9}$$

where CN_{kt} is the total capacity for a resource *k* needed to produce all the scheduled items in the period *t*. The remaining capacity for a resource *k* in the period *t* can be calculated as

$$CR_{kt} = C_{kt} - CN_{kt} \qquad (k = 1,2,...,K). \tag{11}$$

It is obvious that the resources capacities are enough to produce all the scheduled items in a period *t* if the capacity tightness indexes are all less than or equal to one (i.e. $\rho_{kt} \leq 1$ for all *k=1,2,...,K*) at this period. In this case, the shifting procedure moves to the period (*t-1*) and begins to check capacity feasibility in this new period. If in some period *t* the total capacity for a resource needed to produce all the scheduled items in this period is larger than the total available capacity for this resource at the period (i.e. $\rho_{kt} > 1$ for some *k),* infeasibility occurs for this resource *k* and one or more of the scheduled item(s) in this period must be shifted to the previous period(s). If there are more than one such resources, the procedure begins with the resource with the largest tightness index. In order to get a capacity-feasible plan more efficiently, we only consider moving the excessive scheduled production quantity to the period just before this period. In order to reduce as much as possible the possibility of any infeasibility resulted from the shifting procedure for the materials between different stages, the items are shifted according to the decreasing order of the indexes of the scheduled items in this period (i.e., from item *N* downto 1). This is based on the observation that the item with a large index will not use an item with a small index as its components. That's to say, beginning with item *N* and downto item 1, we move the excessive production quantity of one or more scheduled items in this period to its previous period until the capacity-unfeasibility for this resource in this period is eliminated. If there is (are) other infeasible resource(s), we repeat this shifting procedure until all the infeasibility are eliminated. When the moving processes are finished, the modified lot-size plan will be a feasible plan with respect to the capacity constraints. The only infeasibility of this modified lot-size plan may be the backlogging for some items in some periods and this will be considered by the penalty method similar to the cost function (8) for the uncapacitated lot-sizing problem.

The shifting procedure goes successively from period T downto 1 and from the tightest resource to the most flexible one within a period. While capacity infeasibility occurs in some period, move the excessive production of one or more items in this period to the period just before current period to assure capacity feasibility (If current period is the first period, then the production quantity for this item in this period will be partly or completely lost). In this shifting or moving excessive production backwards procedure, the remaining capacities of all the resources in these two successive periods are adjusted through considering the shifted quantity and the possible saved or added setup times (i.e., the setup-consumed capacities). The setup patterns of the shifted items in this two periods will be modified whenever applicable as in memetic algorithm.

It should be noticed that, in this decoding (or called growth function constructing) procedure, local-search is also implicitly incorporated and most constraints in CLSP is implicitly considered and satisfied whenever possible. The detailed implementation and numerical evaluation of this heuristic genetic algorithm can be found in Xie (1995).

5 Summary

Evolutionary algorithm is not an universal prescription to all diseases. Correct prescription to a disease can only be made based on sufficient knowledge about the disease. This paper emphasises the fundamental importance of the ways in which domain-specific knowledge may be systematically exploited and incorporated into evolutionary algorithms, and suggests that domain-specific presentation scheme and domain-specific local search be two basic starting points to combine problem-related knowledge and the general structure of evolutionary algorithms. A heuristic genetic algorithm for capacitated lot-sizing problems is used as an example to demonstrate these principles. We show how evolutionary algorithms can be precisely designed to solve optimisation problem with complicated structure of decision variables and constraints. The authors highly agree with Radcliffe and Surry (1995) that more attention should be focused on how to analytically predict the performance of a specific representation for a specified problem domain.

Acknowledgements

This work has been partially supported by National Natural Science Foundation of China (Project No.???????).

References

P. Afentakis and B. Gavish, 1986. Optimal lot-sizing algorithms for complex product structures. *Operations Research,* 34:237-249.

T. B點k, 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithm*. Oxford University Press, New York.

T. B點k and H. P. Schwefel, 1993. An overview of evolutionary algorithms for parameter optimisation. *Evolutionary Computation*, 1(1):1-24.

R. Clark and V. A. Armentano, 1995. A heuristic for a resource-capacitated multi-stage lot-sizing problem with lead-time. *Journal of Operational Research Society,* 46:1208-1222.

L. Davis, 1991. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

L. J. Fogel, A. J. Owens, and M. J. Walsh, 1966. *Artificial Intelligence Through Simulated Evolution*. Wiley Publishing, New York.

F. Glover, 1986. Future paths for integer programming and links to artificial瑿ntelligence. *Computers and Operations Research*, 13(5):533-549.

D. E. Goldberg, 1989*. Genetic Algorithms in Search, Optimization & Machine Learning*. Addison璚esley, Reading, Mass.

J. H. Holland, 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, 1983. Optimisation by simulated annealing. *Science*, 220(4598):671-680.

J. R. Koza, 1991. Evolving a computer to generate random numbers using the genetic programming paradigm. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp.37-44. Morgan Kaufmann, San Mateo, CA.

G. Laporte, 1992, The Travelling Salesman Problem: An overview of exact and approximate algorithms, *European Journal of Operational Research*, 59, 231-248.

J. Maes, J. O. McClain and L. N. Van Wassenhove, 1991. Multilevel capacitated lot-sizing complexity and LP-based heuristics. *European Journal of Operational Research,* 53:131-148.

M. Pablo and M. G. Norman, 1992. A "memetic" approach for travelling salesman problem: implementation of a computational ecology for combinatorial optimisation on message-passing systems. *Proc. Of Int. Conf. On Parallel Computing and Transputer Applications*. IOS Press, Amsterdam.

N. J. Radcliffe and P. D. Surry, 1994. Formal memetic algorithms. In T. C. Fogarty (ed.), *Evolutionary Computing: AISB Workshop*, pp.1-16. Springer璙erlag.

N. J. Radcliffe and P. D. Surry, 1995. Fundamental limitations on search algorithms: Evolutionary computing in perspective. In J. van Leeuwen (ed.), *Computer Science Today: Recent Trends and Developments*, pp.275-291. Springer璙erlag, New York.

I. Rechenberg, 1973. Evolutionstrategie---Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann璈olzboog, Stuttgart.

I. Rechenberg, 1984. The evolution strategy. a mathematical model of Darwinian evolution. In E. Frehland (ed.), *Synergetics---from Microscopic to Macroscopic Order*, pp.122--132. Springer璙erlag, New York.

Y. Roll and R. Karni, 1991. Multi-item, multi-level lot sizing with an aggregate capacity constraints. *European Journal of Operational Research, 55*:73-87.

P.D. Surry and N. J. Radcliffe, 1996. Formal Algorithms + Formal Representations = Search Strategies. In *Parallel Problem Solving from Nature - PPSN IV*, Springer-Verlag, Berlin.

D. Whitley, V. S. Gordon and K. Mathias, 1994. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature - PPSN III*. Springer-Verlag, Berlin.

D. H. Wolpert and W. G. Macready, 1995. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute.

J. Xie, 1995. Production planning and scheduling: mathematical models and algorithms. PhD thesis, Tsinghua University, Beijing.