第二章 文獻探討

2.1 電子市集

Gartner Group 對電子市集的定義如下,「在網際網路上, 一個提供上下游買賣雙方進行產品或服務的交易中立場所。」它是一個由中立的市集交易管理員在網際網路這個虛擬的平台上, 管理各項有形或無形的商品買賣交易, 並提供支援性服務。Kaplan and Sawhney(2000)將電子市集定義為中立且以網路網路為基礎的仲介商, 著重在某些特定產業的垂直面向, 或是特定產業流程的水平面向。並且以 eHub 的觀念來形容 B2B 電子市集, 認為電子市集匯集了大量的買賣雙方, 藉由自動化交易流程降低成本, 也擴大了買方購買產品與服務的選擇空間, 並使得賣方得以開展新市場和接觸新客戶。

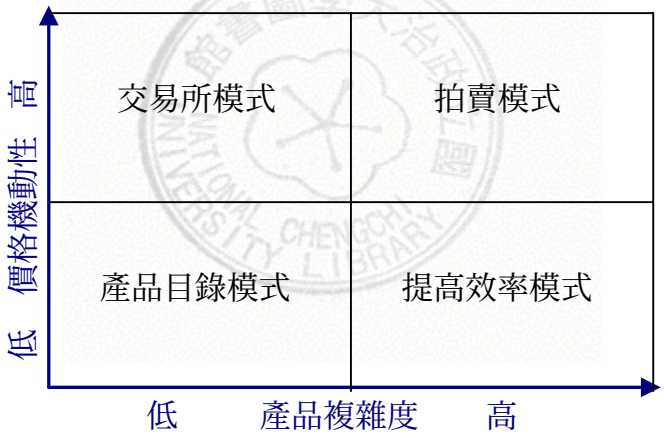


圖 3、電子市集的商業模式

電子市集的分類方式繁多：依主導權來看, 電子市集可分為三類, 買方主導的電子採購(e-procument)、賣方主導的電子配銷(e-distribution)和第三者主導之電子市集；以產業或服務內容項目來看, 可區分成水平與垂直電子交易市集；而依商業模式區分, 有產品目錄入門網站、提高效率模式、拍賣、交易所模式(蔡桂芳、萬洪濤, 民 89)。如圖 3 所示：

一、產品目錄入門網站(Catalog model)

線上目錄(online catalog)可說是最初期的 B2B 電子商務, 基本上就是一種網

路黃頁(yellow pages)，將不同供應商的產品資料分門別類匯聚整合於一網，並且往往必須將商品標準規格化，並將資訊加以整理與加值。藉由大量聚集買賣雙方來創造價值，主要賺取廣告費、會員費和佣金。一般來說，買賣雙方可以在產品目錄入門網站找到對方，但不能進行交易，交易的執行多半還是連到供應商的網站完成。適用於賣賣雙方很分散、交易頻繁但交易量很小和物價穩定的產業。由於交易頻繁且交易量小，為了每次交易而詢價報價太不划算，所以這類模式的售價通常事先定好。

二、提高效率模式(Efficient Commerce Model)

藉由改善買賣雙方既存的商業關係來創造價值。雖然說已有商業往來的買賣雙方共同組成一個交易網路，但是這個交易網路中仍存在很多缺乏效率的因素，例如未完全自動化的採購流程所帶來的人為錯誤、供應鏈資訊缺乏、昂貴的加值網路(VAN)和 EDI 成本。此模式適用於具有以下特性的產業：標準化高、買賣契約已經協議好，對於價格和條約都已經有大致上的共識、買方因為把訂單集合起來以達到更佳的經濟效益。

三、拍賣模式(Auction Model)

傳統的拍賣模式因為交易成本高而無法大型推廣，使固定價格的交易方式大行其道，但由於網路的興起使得交易成本降低，並破除時間空間的限制，讓拍賣模式鹹魚翻身。近年來，網際網路上成立了許多家拍賣網站，其中以 ebay 較為有名。這些網站提供拍賣機制供買賣雙方完成交易，網站上拍賣的商品大部份是價格比較無法確定的商品，例如新上市的商品、二手商品、藝術品、古董等。這些拍賣網站的參與者，不需要和傳統拍賣場一樣共處一室，即可透過網路進行拍賣，但拍賣期間，買方還是要自己經常上線了解拍賣情形並進行喊價，非常麻煩且浪費時間。

拍賣是人類最早發展的交易概念之一，世界各國不約而同發展出自己獨特的拍賣規則，下表列出六種拍賣方式：

表 1、六種拍賣方式

拍賣種類	拍賣方式
英式拍賣（English auction）	最普遍的拍賣方式。拍賣主持人以預定的低價開始，請買方公開喊價，買方喊價越來越高。在單一贏家制度下，由喊價最高者以最高價得標；而在多贏家制度下，則競標者要同時標明購買數量和叫價，由交易總值最高的競標者們得標。適合單一賣方多數買方的環境。
荷式拍賣（Dutch auction）	與英式拍賣相反，拍賣主持人以預定的高價開始，喊價隨著時間越來越低，當買方願意以該價購買，即示意表示，此時拍賣結束，該買方以該價得標，，適合單一賣方多數買方的環境。
日式拍賣(Japanese Auction)	從很低的價位開始，慢慢依照一定的速度提高，直到只剩下一個競標者為止。多位贏家制度時，則由競標者依當時的價位表明購買數量，直到商品全部賣完。
第一價位秘密競標（First-price sealed bid auction）	秘密競標最好的例子就是民間互助會，競標者只有一次機會出價，且互不知道彼此出價。買方將標單密封交給拍賣主持人，拍賣時間結束時，拍賣主持人開標，標價最高者以最高價得標。而在多位贏家制度下，參與競標者必須標明出價與數量，以交易總值最高者得標。適合單一賣方多數買方的環境。
第二價位秘密競標(Vickrey aution)	買方將標單密封交給拍賣主持人，拍賣時間結束時，拍賣主持人開標，標價最高者以次高價得標。適合單一賣方多數買方的環境。
雙重競標（Double auction）	賣方及買方將標單交給拍賣主持人，拍賣時間結束時，拍賣主持人開標，並依類似股票交易所搓合買賣雙方的方法，決定成交的價格及成交的買方和賣方。適合多數賣方多數買方的環境。

(來源：本研究彙整)

最適合採用拍賣交易模式的情況如下：

(1).價格不確定的商品或服務。造成價格不確定的原因為產品很特殊或難以預期市場供需。

(2).過多的存貨。

(3).昂貴或稀有的商品。

(4).需要考慮折舊價值的二手商品。

(5).退貨或過時的存貨。

(6).全新的商品，想測出其市場價值。

拍賣模式搬到網路上，演變出兩種主要的拍賣類型：

(1).順向拍賣(Forward Auction)：賣方在網站上登錄要出售的商品，由買方瀏覽所需商品和出價，最後賣方選擇出價最高的買方。順向拍賣由賣方主控，價格只往上漲，強勢的是賣方，賣方可以隨時終止拍賣，和決定賣給誰。

(2).逆向拍賣(Reverse Auction)：買方在網路上列出自己所需的商品資訊，賣方投標來回應買方的需求，最後買方挑選最理想的賣方成交。逆向拍賣由買方主控，價格通常只往下跌，買方藉著賣方的彼此競爭，而得到最好的價錢和成交條件。

四、交易所模式(Exchange Model)

交易所模式和拍賣模式都屬於機動訂價的交易模式。交易所模式顧名思義，與股票證交所的模式類似。參與的買賣雙方不必有直接的往來關係，僅靠交易所來維繫買方和賣方的關係即可。買賣雙方的身分也非恆久不變，買方也可以是賣方，賣方也可以是買方，反之亦然。所以企業可以利用交易所模式的虛擬商場，來同時進行買進原料與銷售產品。一般來說，任何產也和商品都適用交易所模式，特別是已經標準化的商品，或是不需要作個人化修改的商品，例如汽車零件、飛機零件和紙張等。2000 年 AMR 依照交易所虛擬商場的商業模式、產業經驗、交易量和策略地位，挑出前二十名，排名如下：

表 2、前二十名交易所模式的電子市集

名次	交易所名稱	產業類別
1	Altra Energy Technologies, Inc.	天然氣、液體瓦斯、電力
2	Ventro Corp.(VNTR) (前身為 Chemdex)	生命科學、醫療供給、健康保險、食品
3	CheMatch.com	化學
4	SciQuest.com, Inc(SQST)	化學藥劑與實驗所需的相關產品
5	PlasticsNet.com	塑膠
6	Neoforma.com, Inc.(NEOF)	醫療儀器與產品
7	E-Chemicals, Inc.	化學
8	CheConnect, Inc.	化學
9	Instill Corp.	食品餐飲業
10	The National Transportation Exchanges, Inc.	陸上運輸業
11	Arbinet Communication	通訊產業的網路設備容量
12	E-Transport, inc	海上運輸產業

13	MentalSite	基本金屬與合成金屬
14	E-Steel Corp.	鋼鐵
15	PartMiner, Inc.	電子零件現貨市場
16	BidCom, inc.	建築產業的工程進度管理
17	HoustonStreet.com/Exchange, Inc.	能源產業批發商
18	ECFood.com	食品原料供給鏈
19	SupplierMatket.com	跨產業使用的直接物料，如鋁、不鏽鋼、尼龍等。
20	VerticalMet(VERT)	約有六十個不同產業的垂直虛擬商場

交易所模式的優點和拍賣模式一樣，能讓市場機能自行決定產品的售價。這種讓機動訂價算是最合理的訂價方式了，因為買東西的人認為自己沒有多付，賣東西的人認為自己沒有少賺。根據產業分析師的估算，在 2003 年以前，將有 35% 的網路交易透過交易所模式來完成。

交易進行的程序是：

步驟一：企業對所想購買的商品進行出價(bid)，或對想要出售的商品叫價(ask)。

步驟二：買賣雙方可看到彼此的出價和叫價，但是不知道對方是誰。

步驟三：透過出價和叫價的即時互動來達成交易。

步驟四：有付款和運貨等服務使交易更完善。有時有第三者提供信用稽核、品質保證和履行訂單等支援功能。

2.2 協商

電子網路上的消費購買行為與傳統商業行為有相似之處，大致分成六個階段(Maes, Guttman and Moukas 1999) ：需求確認階段、商品仲介階段、廠商仲介階段、協商階段、購買與遞送階段、及服務與評估階段。其中協商階段即為本論文所研究的重點。

2.2.1 協商的定義

協商是一個在不完全資訊的環境下，多個自利(self-interested)的代理人之間競合的決策制定過程(Rahwan et. al. 2002) 。協商是雙方或多方共同的決策流程，協商者將互相衝突的需求以言語表述，然後透過讓步或搜尋替代方案來達到協議(Kowalczyk and Bui 2000)。因此本研究將協商歸納定義為：協商是一個決策制定過程，在不完全資訊的環境下，兩方或兩方以上聯合搜尋可行解的範圍，透過讓步和提出替代方案來慢慢解決衝突，最後達成一致的協議(consensus)。

而在電子商務的環境下，協商的目的在於讓交易夥伴之間對電子交易產品或服務的條件達成協議。協商的雙方通常不了解對方的偏好、限制和目標，或僅得知有限的資訊。協商雙方以提案(offer)的方式來作為訊息交換，一個提案包含一些協商議題(issue)的值，例如價格、數量、運送時間等，而這些協商議題就是在協商過程中不斷調整以達到雙方協議(Alem et. al. 2000)。

2.2.2 協商的分類

協商依參與者數目可分為：

(1).雙邊協商(Bilateral negotiation)：只有兩方參與協商，一對一的協商通常用於工作配置問題(task reallocation problem)和資源分配問題(resource reallocation problem)，有限的資源可能是時間、完成某些工作的特殊資源。

(2).多邊協商(Multilateral negotiation)：超過兩方以上參與協商，又可分為一對多和多對多的協商。一對多的協商通常用於拍賣；而多對多協商是複雜的協商情況，為了降低其複雜性，有學者提出多對多的協商可由許多一對一或一對多的協商所組成(Rahwan et. al. 2002) ，過度簡化了多對多協商的問題，因為多對多的協商問題不只是議題，還必須考慮到其他參與者對整個協商的影響力。

協商依協商屬性的數目來可分為(Guttman and Maes 1998；Lewicki et. al. 1997；Mumpower, 1991) ：

(1).競爭性的協商(Competitive Negotiation)：解決兩或兩方以上對單一互斥目標(mutually exclusive goal)衝突的決策制定過程。即是以單一屬性來協商，協商雙方有著完全相反的目標。例如消費者在購買商品時，往往只考慮價格因素，買方想要價格越低越好，賣方想要價格越高越好，一方贏得的值即是另一方所輸去的。因此，單一議題的協商又稱為分配型協商(Distributive Negotiation)，因為協商雙方僅就一個議題來討論，一方獲得的利益，另一方必定有所損失。在賽局理論中，又稱為零合遊戲(Zero-sum game)或輸贏遊戲(win-lost game)。

(2).合作性的協商(Cooperative Negotiation)：解決兩方或兩方以上對多個不獨立、非完全互斥目標(multiple interdependent, but non-mutually exclusive goals)的衝突的決策制定過程。因為有多個協商屬性可以作為協商的 trade-off，因此可以找出雙贏的解決方案。例如大部分的消費者考慮的不只是價格因素，可能還包括付款方式、交貨天數、保固期限、送貨方式等因素，此時買方與店家的協商就成為多議題協商。多議題協商的彈性遠比單議題來的大，買賣雙方能夠以互補的方式各在自己較不重視的議題上讓步，最後以互利的方式來達成交易，因為在某一議題上的讓步，可能可以在別的議題上獲得補償，最後有可能雙方都贏。因此這種型態的協商也稱為整合性協商(Integrative Negotiation)，在賽局理論中，又被稱為非零合遊戲(Non Zero-sum game)或雙贏遊戲(win-win game)。

在真實交易情況中，就單一議題協商來說，會有兩種極端情形：合作或是不合作。但是為了處理較複雜的商品或服務，多屬性協商也是需要的，而且經由協商各方對不同屬性的偏好度，較能達成雙贏的結果。因此本研究所提出的多對多多屬性協商模式即是整合多邊協商和合作性的協商，結合兩者優點而成。

2.3 自動化協商

2.3.1 自動化協商系統和協商支援系統

一、自動化協商系統

其主要目的是將人類的協商過程自動化。協商可視為一個搜尋的過程，買賣雙方在成交空間中尋找最佳的方案，在過程中包含參與者之間互相的往來出價。依據不同的商業交易種類，協商議題、期間、及複雜度將會有所差異。再者，他也需要一些屬性，諸如價格、送貨情況、保證、售後服務、及其他加值服務。雖然在人類文明史中，人類協商很早就出現，但是不盡然能適合在全球性或線上市集上，所以自動化協商是需要的。軟體代理人在電子商務中扮演一個重要的角色，如資訊的擷取：尋找商品、商務資訊、比價、及提供客製化的建議。人類偏好能線上立即互動，彷彿有生命的軟體代理人。人類協商相對來說較為緩慢，無法總是找出最佳解，而且受限於文化、自我意識、情緒或是自尊。而實際上，人類也無法勝任全球性 24 小時的線上交易環境。自動化協商能保證快速、一致性、及消除人類的錯誤，並提供全天候的線上協商能力。

談判代理機制最早期是美國麻省理工學院媒體實驗室中所研發的 Kasbah(Chavez and Maes 1996)，是探討零售市場的協商機制。Kasbah 是一個集中的代理人電子市集，當買方或賣方想要交易時，便創造一個代理人派去集中式代理人電子市集，系統中每一個零售商可以有一個自己的代理程式，放在網路上所謂的虛擬市場之上。每一個消費者如果想買商品時，可以透過代理程式到虛擬市場中代替消費者去做詢價與比較的工作，使用最有效率的方法來買到最便宜的商品。但是 Kasbah 的缺點為僅以價格作為協商考量，且策略運用的考慮過於簡單。後來美國密西根大學發展出拍賣伺服器 AuctionBot，可以選擇拍賣的種類，然後再指定所需的參數，如交易時限和保留價格，所以 AuctionBot 仍然只考慮價格。Tete-a-Tete 提供了多屬性的協商，可以就產品保證、送貨時間與方式、服務契約、退貨政策、贈品服務及其他交易的項目進行協調，算是比較完整的談判機制，但是只考慮一對一或一對多的協商(Maes, P., Guttman, R. H. and Moukas, A. G., 1999) 。

二、協商支援系統(Negotiation Support System, NSS)

主要功能是輔助使用者，在協商過程中能有效提供參考資訊，讓使用者做出正確的決策，縮短協商時間，或是提升協商結果的效率。就電子交易的應用來說，如網路搜尋比價、拍賣、競標等等功能的網站代理程式，就是能幫助使用者節省時間和成本的機制。一個考量實際情況的協商支援系統，需要能處理不確定及遺失的資料，而且僅是輔助協商者做決策(張婉華，民 90)，雖然人工智慧的應用廣泛，但是尚未有真正能模擬人類大腦的機制出現，人類複雜精密的思考決策能力有如一台超級電腦，若以自動化程式來執行人類決策，並不能有效處理較複雜的協商行為，而且自動協商或經由軟體代理人仲介的協商結果，並不能使全數協商參與者滿意，甚至要做事後調整來做後協商的動作，所以若協商支援系統的預測功能強大，並提供許多圖表及重要資訊，以一個 well skill 及具備協商 pre knowledge 的協商者來說，將能有效提昇協商時的效率。

此兩種系統的使用時機，必須依照協商性質及環境的不同而有所取捨，各有優劣。當協商議題多時(>4)，在調整出價上較為複雜，也使得使用者在搜尋及比較提案效用的動作上，也是極為耗時且費力，協商時間更因此而延長。這時使用自動化協商比較方便。但是自動協商系統存在一些固有的缺點，如協商策略固定，而且可用的有效策略不多，再者，每個提案對雙方的效用不一定是相等的，即使是落於效率前緣及公平的成交點上，若自動協商的成交結果有一方不滿意，而提出「後協商」(post-negotiation)的要求，那自動媒合機制的功效及美意就大打折扣。另外一個在協商支援系統方面的固有特點是，協商需要教育訓練，一位缺乏協商經驗的人與一位擅長協商的人協商，那結果是可以預期的，就比如打官司需要聘請律師協助是一樣的，有一個技術高明的代理人(Agent)幫助協商，甚至全權處理複雜的協商行為，在某些情況下是需要的。

2.3.2 自動協商技術

以下介紹一些自動協商的技術，包括窮舉法建立提案表、基因演算法、模糊理論、賽局理論和限制滿足問題等。

一、窮舉法建立提案表：

根據協商者的喜好，將所有可能的協商提案計算出放入提案表，並以效用函數來計算每個提案的效用，提案表依據提案本身效用由大到小排列，然後依據這提案表進行協商，從效用最大的提案開始，在協商的每一回合逐步往下找出一個適當的提案送給對方。但是必須假設提案組合總數仍屬於可處理(窮舉)的範圍內，然而實際應用上，當議題數目增加，變動單位降低時，提案個數就會大幅增加。張婉華(民 90)便是用此方法來建立一對一多屬性的協商；而陳廷豪(民 91)也是利用同樣來方法來達成多屬性的協商。

二、基因演算法(Genetic Algorithm)：

由於協商結果的解空間之廣大，在無法以窮舉的方式來求解的情況下，以漸進方式的求取較佳解的基因演算法便成為眾多研究者的工具(朱宏揚，民 89)。這種方式大多首先針對所有可能的提案(offer)訂立合適函數(fitness function)，並在協商一回合之後，利用複製(reproduction)、交配(crossover)及突變(mutation)等運算來產生下一次(代)可能的候選提案。如此在一定的世代之後，可能就有機會找到一個較佳提案並達成交易。這種方法雖然在文獻中展現不錯的實驗結果，但是由於運作過程中需要大量的經驗法則(heuristics)來調整相關參數(例如母體大小、突變率、交配率、繁衍代數)，使得結果難以預測。明顯的缺點有：(1)無法保證協商結果是否會符合效率、共同效用最大和公平等三項指標(此三項指標將會在後續文獻探討中進一步詳述)，(2)協商初始提案的訂定原本不易，不良的初始提案將可能影像最後的結果，然而基因演算法的第一代多以亂數隨機產生或假設分析的方式來產生。

三、模糊理論：

模糊理論提供了另一種決策模式，它可以表示人類在描述事物上的不確定性。郭建男(民 90)以模糊推論算出兩個協商屬性構成的提案之效用，使用者不必設定權重，讓一個買方可以評估比較多個賣方的提案，其實驗發現用模糊推論與簡單加權法進行比較，發現模糊推論法與簡單加權法所得的結論很接近，意即模糊推論在多屬性決策上有其適用性，也提供了另一種不同於須設定權重的決策模

式。但其缺點是，2 個屬性 5 個模糊語意所構成的模糊規則為(5*5)，如果為三個屬性則為(5*5*5)，複雜度升高，造成使用者設定的麻煩。

四、賽局理論(Game Theory)：

在買賣雙方均為理性玩家的假設下，列出所有雙方可能使用的策略(Strategies)，並分析當雙方所有的策略組合的效用(payoff)，以提供協商參與者建議，這種做法雖然可以達到最佳解，但在現實的情況下，策略之數目龐大而難以窮舉，更何況要個個加以配對分析，因此實際應用有其困難之處。

五、限制滿足問題(Constraint Satisfaction Problem)：

將協商問題解釋為限制滿足問題，求解產生協商提案，並搭配多屬性效用理論作為決策分析工具，來解決一對一或一對多的多屬性協商，此法為多位學者所採用(Guttman and Maes 1998；Kowalczyk and Bui 2000；Rahwan et. al. 2002) 。本研究便是採用此方法來建置多對多的多屬性協商，限制滿足問題和多屬性效用理論的詳細介紹在 2.4 跟 2.5。

2.3.3 協商問題結構

根據 Mumpower 的定義：協商問題結構(Structures of Negotiation Problems)就是所有協商者的可行解空間的聯集，以及此聯集區域的效率前緣，協商者如何定義可行解的效用，也會影響整個結構的改變。就雙邊多議題協商類型來說，根據協商者的喜好，可以找到雙方共同可行解的聯集，將這個聯集中的所有解帶入我方的效用函數得到 Y 軸，再帶入對方的效用函數得到 X 軸，就可以畫出該次協商的問題結構(張婉華，民 90)。如下圖所示，圖中的每一個點代表一個提案，所在的位置由其相對於我方與對方的效用而定。落在第一象限的解為共同的可行解，使我方和對方的效用都大於 0 的效用值，照理說應該是雙方共同接受的區域，只是滿不滿意的問題。有的時候對方的提案帶入我方的效用函數得到大於 0 的效用值，但是提案中的議題值不一定在我方可接受的範圍內，我方仍是不能接受。所以能使雙方效用都大於 0 的提案，還必須讓提案中的每個議題值都符合雙方各自設定的議題範圍內，才是共同的可行解。

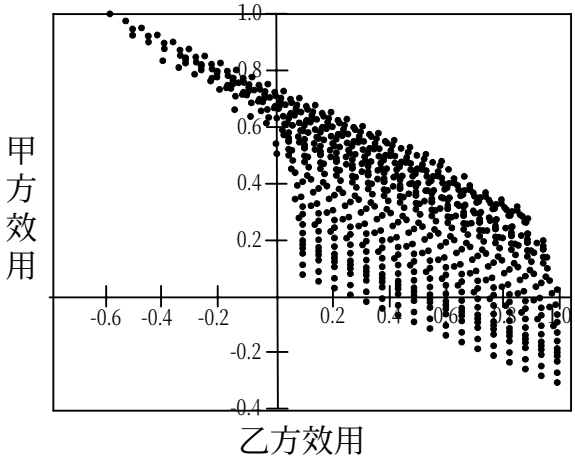


圖 4、甲乙兩方協商的問題結構

協商雙方對每個議題的重視程度以及議題效用函數的變化是呈現直線、遞增或遞減型態，會改變協商問題的結構，而協商雙方的第一次提案以及對每個議題的最高最低出價也會影響協商問題結構。「整合型協商」就是在尋找滿足這種多維空間限制區域內的共同可行解。所以，協商的目的是要讓雙方進入這個空間以達成協議，而一個有效率的協商不只是能使雙方達成協議，還能使雙方有最高的效用，也就是尋找空間中的最佳解。

2.3.4 協商結果的判別

協商成交點位於協商問題結構圖中的位置不同，有不同的協商意義，衡量協商結果優劣有以下三種指標(張婉華，民 90；Mumpower 1991)：效率、共同效用最大、公平。好的協商結果至少應該符合效率指標，但不一定得符合共同效用最大和公平，會因雙方是競爭或合作對象而異。但是如果能完全符合此三項指標，照常裡推斷應該是使雙方都能滿意的最佳協商結果。

1. 效率(Efficiency)

由協商雙方共同可行解所組成的右上方的邊緣線，即是效率前緣(Efficient Frontier)，凡是落在效率前緣上的解，就無法找到另一個解，可使雙方的效用都增加，或是一方不變另一方增加，也就是雙方滿意度最大的解，或稱柏拉圖最佳值(Pareto Optimal)。如果不是落在效率前緣上，則表示在不減少任何一方利益的

狀況下，有更佳的解存在。協商最後的協議落是落在效率前緣上就表示此協商結果是有效率的，但是在這條效率前緣上的每個解對雙方的效用來說不一定是公平的。

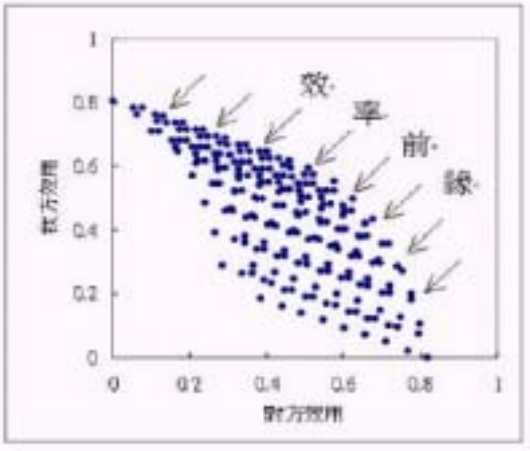


圖 5、買賣雙方共同可行解區域

2. 共同效用最大化(Joint Utility Maximization)

協商達成協議後，成交提案有相對於雙方的效用值，將雙方的效用值相加，就是共同效用。共同效用最高的點一定落在效率前緣，所以共同效用越高，成交點越逼近效率前緣，所以如果協商結果沒有落在效率前緣，則表示雙方的共同效用還有提高的空間。然而在效率前緣上的點不一定能夠使雙方效用最大，因此在效率前緣上的每個點也可以再互相比較共同效用，但是共同效用最大的解不一定具有公平性。

3. 公平性(Equality)

雙方所獲得的報酬(payoff)是否公平一致。協商結果的效用若是一方高於另外一方，表示協商的結果分配不均，一方獲利太高，另一方有損失，則協商的結果就不具公平性。以下圖為例，在效率前緣上最接近 45 度線的解，就是最公平的解，若與效率前緣相交，則協商結果就是最有效率的解。

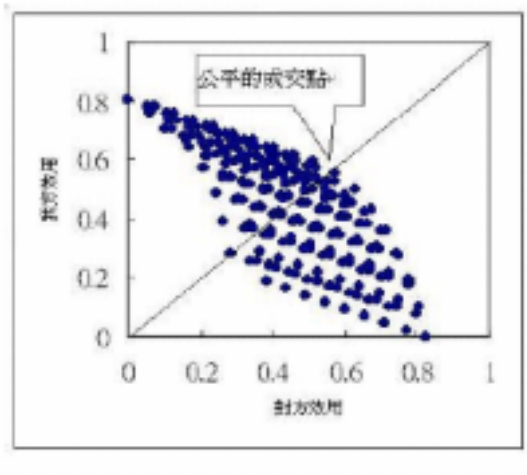


圖 6、公平的成交點

由以上三點衡量指標來看，協商的最佳解一定要落在效率前緣上，但是在效率前緣上，最大共同效用的解和雙方效用相等的解未必是同一點。因此要同時滿足這兩個衡量指標不容易。所以最佳解並不容易定義，只能說在效率前緣上能夠盡量滿足最大共同效用和雙方效用最小差異的解就是最佳解。

2.3.5 協商策略

自動化協商的相關研究主題中，最重要的首推如何訂立有效的策略來處理初始提案(Initial proposal)、讓步(Concession)策略與防弊等主題。以下介紹一些應用在自動協商的策略。

一、集合式提案(Grouping Proposals)

這是由張婉華(民 90)所提出的提案策略，立論是在未能確知對方喜好的情況下，我方並不知道在這段讓步幅度間，哪一個提案會使對方效用最高，也就很容易錯過使對方效用較好的提案，因此，在我方已經決定的讓步幅度下，將效用介於這個範圍內的所有提案提出給對方，讓對方在這些提案中找到效用最高的提案。使得我方的讓步一定能增加對方的效用，因而能解決無法預測對方喜好而提出更差提案的問題，並使協商結果一定能落在效率前緣上。

相對於集合式提案為單一提案，單一提案為一次只提一個提案給對方，對方

沒有選擇我方提案的空間。但單一提案的協商結果不一定能落在效率前緣上。

二、讓步策略

協商策略是指協商參與者在每回合中己方效用讓步的幅度, 在自動協商機制下, 需要設定此參數給代理軟體參與協商。在無法得知對方的偏好和效用函數的情況下, 有幾種退讓策略(張婉華, 民 90)：

1. 固定退讓策略(Fixed concession)：

每次的讓步幅度都一樣, 固定不變。

2. 遞減退讓策略：

一開始做大幅度的退讓, 但隨著協商回合增加, 讓步幅度越來越小, 這種策略有逐漸強硬的趨勢。

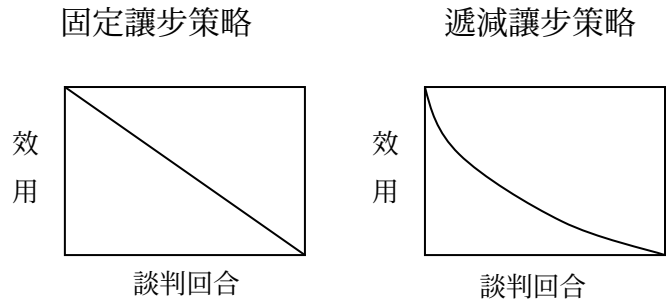


圖 7、固定讓步策略和遞減讓步策略

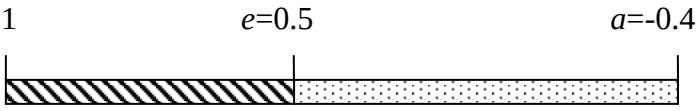
3. 設定期望效用的退讓策略：

協商之前必須決定我方的立場, 此立場可能是很急著成交, 或是想獲得較大的利潤, 因此先讓使用者設定一個期望成交效用, 這個效用介於 0 與 1 之間, 當期望效用越大, 代表使用者越強硬。公式為：

$$\Delta = (X'_{i-1} - X'_i) \times \frac{1-e}{1-a}$$

Δ：我方下一回合的讓步幅度, 也就是相鄰兩回合的效用差。
i：協商回合, i=1…n, n 是最終回合。
$(Y_{i-1} - Y_i)$：對方的讓步

$(X'_{i-1} - X'_i)$：對方給予我方的效用
a：對方第一次提案換算為我方的效用，a<1
e：我方的期望成交效用，e 在 0 到 1 的範圍內



1 是我方第一次提案的效用
-0.4 是對方第一次提案使我方的到的效用
0.5 是我方的期望效用

圖 8、設定期望效用退讓策略之舉例圖

以上圖為例，說明此策略的做法，我方希望最後在 0.5 的效用成交，我方必須在退讓到效用為 0.5 時，對方也給我方 0.9 的效用距離，雙方才會在我方效用 0.5 的地方成交。所以，將對方每次給予我方多出的效用乘上(0.5/0.9)就是我方每次的讓步幅度。

4. 互利的讓步策略：

因為雙方都無法得知另一方的真實讓步幅度，只能依據對方給予我方的效用值來決定要讓步多少，依照己方的效用函數來計算另一方給予的效用，並且需要設定讓步門檻及希望成交的效用值，使協商籌碼不致過度使用而浪費掉。對方所給予我方的效用有時是很大的值，若是大於門檻值，則退讓程度就等於門檻值，若是小於門檻值，則與對方作同樣大小的退讓。例如，我方的門檻值是 0.1，當對方給予我方的效用是 0.05 時，我方就讓步 0.05；當對方給我方的效用是 0.15 時，我方就只讓步 0.1。可表達為公式：

$$\Delta = (X'_{i-1} - X'_i), \text{ if } (\Delta > t) \text{ then } \Delta = t$$

$(X'_{i-1} - X'_i)$：對方給予我方的效用
t：退讓的門檻值

在張婉華的實驗中，可以看出互利的讓步策略是優於固定讓步策略和遞減的

讓步策略。

三、更佳協議策略(Better deal strategy)：

為協商者找到滿意的協議但仍繼續協商的策略, 當協商者得到一個可以接受的提案之後仍繼續協商, 以希望獲得更好的提案。朱宏揚(民 89)實驗發現更佳協議策略所得到的效用會高於滿意即成交的方式。此策略用在協商雙方一次只提一個提案給對方的時候, 由於不知道對方喜好的情況下, 達成協議的點大部分都未落在效率前緣上, 因此達成第一次協議後的雙方可以繼續協商, 以找出可以增加雙方效用的協議。

2.4 限制滿足問題

2.4.1 限制滿足問題

限制滿足問題（Constraint Satisfaction Problem, CSP）的研究在人工智慧領域中, 已有一段很長的歷史, 在人工智慧上的許多應用問題可以轉化成限制滿足問題。許多問題都可以被視為限制滿足問題, 例如排程(scheduling)、規劃(planning)、配置(configuration)、(machine vision problems), 只要目的是在找出一個能滿足所給定條件之配置（configuration）的這類型問題, 都稱為限制滿足問題, 一個限制滿足問題又可以稱為一個限制網路(constraint network)。一個限制滿足問題是由變數(variables)、每個變數各自的值域(domains)、和變數之間關係的限制(constraints)所組成。限制滿足問題的目的在找出一組能滿足所有相關限制條件之問題變數值的組合。儘管限制滿足問題的定義很簡單, 但是許多人工智慧上的問題都能表示成限制滿足問題。一個典型的限制滿足問題即是 N 皇后的問題。

限制滿足問題正式的定義為：有 m 個變數 $X_1,X_2,..., X_m$, $X=\{X_1,X_2,..., X_m\}$, 其各自的值域為 $D_1,D_2,.., D_m$, 並有一組限制。若集合 $S \subseteq \{X_1, X_2,.., X_m\}$, 而限制 C_S 以 S 為範疇(scope), $S = \{X_{i_1}, X_{i_2},..., X_{i_r}\}$, 則 C_S 為所牽涉變數之值域的笛卡

兒乘集(Cartesian product)的子集合，即$C_S \subseteq D_{i_1} \times D_{i_2} \times \cdots \times D_{i_r}$。找到各變數的值並且滿足所有限制，如此便是解決一個 CSP 的問題。

一個限制以述詞(predicate)來定義，該述詞的形式並沒有限制，可以是數學式、邏輯公式或是以一組變數來定義的任何關係式，特別的是可以組合一組變數的值來當做限制，例如(x, y)={(0,2),(2,4),(3,6)}，這種限制稱作 nogood。另外，限制可以分成硬式限制(hard constraint)和軟式限制(soft constraint)，所謂的硬式限制就是等式限制式，相對的，軟式限制是不等式限制。

一個變數從它的值域中被指派一個值給它時，則該變數是已被舉例(instantiated)，否則就是未被舉例(uninstantiated)。$x_i = a$ 或 (x_i, a) 就是將 a 指派給變數 x_i，而 a 為變數 x_i 值域 D_i 中的一個值。對 X 子集合的一個「部分舉例(partial instantiation or partial assignment)」是由一個排序且成對的變數和值的組合 $((x_1, a_1),..,(x_i, a_i))$，可縮寫為 $(a_1,..,a_i)$ 或變數順序已知的 $\vec{a}_i$。如果 $\vec{a}_i$ 滿足所有 $x_1,x_2,..,x_i$ 的相關限制，那麼 $\vec{a}_i$ 是「一致的(consistent)」。而一個「一致的部分舉例(consistent partial instantiation)」又可以說是「部份答案(partial solution)」。一個解(solution)就是一個所有變數都一致(consistent)的舉例(instantiation)。

有限範圍的限制滿足問題(Finite-domain CSPs)是一種限制滿足問題，其變數個數是有限的，變數各對應到一個有限的值域，並且有一組限制來定義變數之間的關係。

2.4.2 限制滿足問題演算法

因為限制滿足問題一般是 NP-complete，所以無可避免的要用嘗試錯誤法(trial-and-error)來找出替代方案。許多學者陸續提出解答限制滿足問題的演算法，這些演算法大致上可分成兩大類。第一種為搜尋演算法，第二類為一致性（consistency）演算法。搜尋演算法用於找到限制滿足問題的答案。而一致性演算法是前置處理的演算法，通常搭配在搜尋演算法之前執行，以減少不必要的搜尋。搜尋演算法又可以分為回溯演算法(backtracking)和反覆改善演算法(Iterative Improvement)兩種。

一、搜尋(Search)演算法

(1).回溯演算法(backtracking)

搜尋可以代表大部分以猜測進行的演算法，而猜測可能是根據某些經驗法則(heuristic)來增加效率。良好的猜測可以讓新的狀態更接近目標，如果是不良的猜測則會遠離目標，這樣不良的猜測必須被撤回(retract)，並繼續作出新的猜測。

一個解決 CSP 的簡單且基本的演算法為回溯(backtracking, BT)，是傳統限制滿足問題的標準技術，其作法是在搜尋圖上以深度優先(depth-first)的方式進行循序搜尋。初始時部分答案(partial solution)是一個空集合，然後選擇一個變數給予一個值，如此一個一個的加入部分答案中，慢慢擴大目前的部分答案，直到完成一個完整答案(complete solution)。變數要加入到部分答案中，其變數值須滿足已存在部分答案中之變數的所有相關限制條件。如此反覆加入，直到部分答案變成完整的答案為止。而當一個變數找不到合理值時，則上一次給予變數值的動作將被撤回(又稱 backtrack)。簡而言之，回溯演算法有三階段：

1. 前進階段(forward phase)：先選擇一個變數。

2. 指派值階段(assign value phase)：為該變數選擇一個滿足所有相關限制的一致的值(consistent value)。

3. 後退階段(backward phase)：如果該變數找不到一致的值則撤回到上一次的變數。反之，則繼續前進。

```
procedure BACKTRACKING
Input: A constraint network with variables {x_1, ..., x_n} and domains
{D_1, ..., D_n}.
Output: Either a solution, or notification that the network is inconsistent.
    i ← 1                                   (initialize variable counter)
    D'_i ← D_i                              (copy domain)
    while 1 ≤ i ≤ n
        instantiate x_i ← SELECTVALUE
        if x_i is null                      (no value was returned)
            i ← i − 1                       (backtrack)
        else
            i ← i + 1                       (step forward)
            D'_i ← D_i
    end while
    if i = 0
        return "inconsistent"
    else
        return instantiated values of {x_1, ..., x_n}
end procedure

procedure SELECTVALUE
    while D'_i is not empty
        select an arbitrary element a ∈ D'_i, and remove a from D'_i
        if (x_i, a) is consistent with ā_{i−1}
            return a
    end while
    return null                             (no consistent value)
end procedure
```

圖 9、回溯演算法

雖然回溯演算法很簡單，但在最差的情況下，回溯演算法需要花費指數時間(exponential time)和線性空間(linear space)。然而有許多考量可以用來增加搜尋的效率。例如變數選取的順序和值域值(domain value)選取的順序會影響到搜尋效率。更有許多學者提出經驗法則（heuristic）來增進搜尋的效率。

在變數選取順序的決定上，通常採用最先失敗原則（first-fail principle）的經驗法則，做法是優先選取限制強度最強的變數，但是如何去量測變數的限制強度呢？最簡單的方法是計算與變數相關的限制述詞(constraint predicate)數目，但是此法無法計算出在部分答案的現有狀態下變數的限制強度。比較好的方法是計算已在部分答案中之變數的連結（link）數量，這是一種動態的量測。而較精確的量測是計算變數和部分答案達成一致性的變數值數目，此法稱為前推式檢查

（forward-checking），其在回溯搜尋的過程中，每一個變數保有一個串列。此串列記錄著變數和現有部分答案達成一致性的區域值。此法有早期偵知失敗的優點。

而在值域值選取順序的決定上，最小衝突（min-conflict）經驗法則被認為是一個非常有效率的經驗法則。在最小衝突回溯演算法中，每一個變數都有其暫時性的初始值。當變數被加入部分答案中時，其暫時性的初始值會被修正。被修正的值必須滿足已存在部分答案中之變數的所有相關限制條件。若同時存在多個值滿足已存在部分答案中之變數的所有相關限制條件時，我們選擇這些值中滿足其他未加入部分答案之變數的相關條件最多的那一個。

另外一個增進回溯演算法效率的方法是相依指向的倒回(dependency-directed)回溯。主要是識別出引起回溯的失敗原因，然後演算法能夠只回到相關的決定點。

(2)反覆改善(Iterative Improvement)

所有的變數給予暫時的值，一個有缺陷的解可以透過爬山搜尋法(hill-climbing search)來改善，然而爬山搜尋法天生的缺點就是會限於局部最佳化或局部最低點(local-minimal)而無法跳脫，局部之最低點表示有違反了一些限制條件，但是限制違反的數目無法以改變任何單一個變數來使其減少的狀態。。因此後續研究便提出各種如何跳出局部最佳化的方法，例如 GSAT 演算法和突破演算法(Breakout Algorithm)。在反覆改善的演算法中，一個錯誤可以不必用徹底的搜尋（exhaustive search）就能修正。因為相同的變數可以一再的被修正。雖然反覆改善的演算法是有效率的，但演算法的完整性無法保證。

二、一致性演算法(consistency algorithm)

一致性演算法有多種俗稱(constraint propagation 或 consistency enforcing)，藉由推論出新的限制、收縮目前限制和移除變數值域中不合理的值，而可將某限制滿足問題轉變成為更明確清楚的相同問題。

一致性演算法以ｋ-一致性(ｋ-consistency)來分類。基本上有節點-一致性(node-consistency)、弧線-一致性(arc-consistency)和路徑-一致性（path-consistency）三種實際可行的演算法。ｋ-一致性演算法將所給的問題轉換成等效的ｋ-一致性

問題。一個限制滿足問題滿足下列條件就是有 k-一致性的特性：

一對任何 k - 1 個變數，給定任何滿足這些變數所有限制條件的任意例子（instantiation），有可能去找到任意第 k 個變數的例子，使得這 k 個變數值滿足它們之間的所有相關的限制條件。

舉例來說，一個 CSP 如果滿足其所有單元限制(unary constraint)，則該 CSP 是 1-consistenct；一個 CSP 如果滿足其所有二元限制(binary constraint)，則該 CSP 是 2-consistent(又稱 arc-consistent)。

2.4.3 協商和限制滿足問題的關係

協商的問題可被視為限制滿足問題(Kowalczyk and Bui , 2000)。

一、變數方面：

$x^j = \{x_i^j\}, i = 1,..,n_j$ 是第 j 個協商者的決策變數集合。**X**則是p個協商者的決策變數集合，因此 $X = \bigcup_{j=1,..,p} x^j$。而 $x = \bigcap_{j=1,..,p} x^j$，$x$不能為空且$x \subseteq X$。本研究假設所有協商者的決策變數相同，即所有協商者的協商議題相同，因此 x=X。

二、值域方面：

值域 D_i^j 包含 x_i^j 可行值的集合，$x_i^j \in D_i^j$，而**D**={ D_i^j } for **X**={ x_i^j }, i=1,..,n_j, j=1,.., p 為所有協商者的決策變數所對應之值域集合。每個協商者對於相同的協商議題可有不同的值域。

三、限制方面：

$C^j = \{C_k^j\}, k = 1,..,m_j, j = 1,..,p$ 是第 j 個協商者的限制，$C = \bigcup_{j=1,..,p} C^j$ 包含所有協商者的限制。若一個限制C_k^j 牽涉到變數x^j 的子集合$\{\chi_{i1}^j,...,\chi_{il}^j\} \subseteq x^j$，則可表示為$C_k^j(\chi_{i1}^j,...,\chi_{il}^j)$。因此$C^j$ 和C分別可以用 $C^j(x^j) = \wedge_{k=1,..,m_j} C_k^j(x^j)$ 和

$C(\mathrm{X}) = \underset{j=1,..,p}{\wedge} C^j(x^j)$ 來表示。

C^j 便是各協商者所偏好的解(individual areas of interest)；C 是協商者之間可能的共同解(common area of interest)。而協商的目的就是從可能的共同解中找出某個解。下圖即表示一個以限制表達的協商問題，包含 a、b 兩個協商者，$C^a(x)$ 和 $C^b(x)$分別代表 a、b 雙方的偏好的解空間，而 x*為從 a、b 兩者偏好解空間的交集中所產生的解。而兩者所共同可能的解空間以兩方限制的交集(conjunctive combination) $C(x) = C^a(x) \wedge C^b(x)$ 來表示。

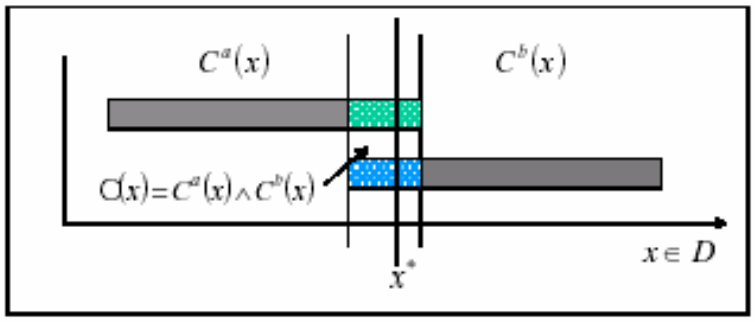


圖 10、以限制為基礎的協商模式，牽涉 a 和 b 協商雙方

協商是一個在不完全資訊的環境下，多個自利的(self-interested)的代理人之間競合的決策制定過程。代理人之間僅以提案(offer)來溝通，並不知道其他代理人的偏好、限制和效用等私密資訊。所謂提案即是一個代理人所偏好的完整方案。因此協商是一個不斷反覆的過程：衡量評估對方所提提案、更新可行解的範圍、並根據協商策略提出新提案。(Rahwan et. al., 2002)

每個協商者一開始協商時會先從本身的解空間中提出對自己最有利的提案，假如該提案不被對方所接受，則繼續提出因應提案(counter-offer)以朝向能達成協議的方向移動，在協商過程中，藉由刪除不合理不可行的解，讓每個協商者的可行解空間不斷縮小直到達到協議為止(Kowalczyk and Bui, 2001)。

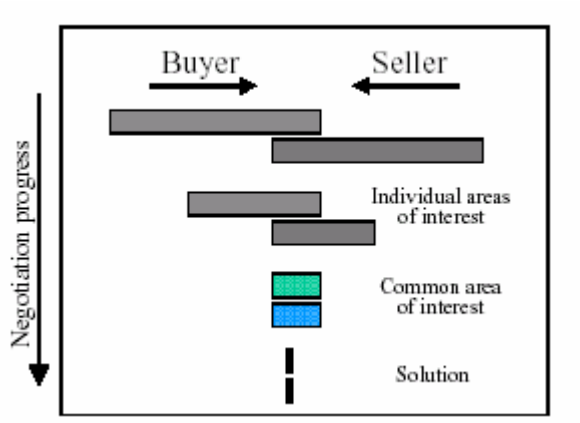


圖 11、買賣雙方協商

2.5 多屬性效用理論

多屬性效用理論(Multi-attribute Utility Theory, MAUT)是將評估問題簡化成一屬性效用函數，以量化的效用來分析決策問題的偏好，主要是運用在多個替代方案選擇；多屬性效用函數即為衡量決策者內心滿足程度的函數。透過效用分析的方法，將評估一多屬性效用函數複雜的問題，簡化成評估一系列單一屬性的效用函數，求取價值函數及權重，再計算績效總表現，也就是利用各屬性的偏好值的組合，合併為對整體方案的偏好。準則、權重和表現績效間的關係呈簡單函數關係。模式組成必須考慮四大部分：

1.屬性的決定：

屬性的決定是多屬性模式的第一步，有以下原則：互斥性(各屬性間沒有很大的共同點)、周延性(所選擇的屬性能清楚涵蓋方案的特徵)、明確性(屬性要明確、易於瞭解)、決定性(每個屬性有決定性的影響力)。

2.價值函數的決定：

價值函數衡量的方法可分為兩種，一為直接法，由決策者自決定其偏好值對各屬性的不同水準加以評定；二是間接法，由決策者對不同的屬性組合加以評分或評定次序， 再經由研究者計算出偏好值，一般就直接法與間接法比較來說，直接法容易被決策者所理解，而間接法在處理上較繁雜，且決策者不容易回答問題。

3.模式的函數型態：

相加性模式有線性加法模式、高階加法模式、交互作用加法模式等；相乘性模式有聯合模式、解離模式等。目前發展較多的是相加性模式，以可衡量加法價值函數理論(Measurable Additive Value Theory, MAVT)為主。

相加模式與相乘模式分析比較如下：

- 當方案的整體效用與屬性的偏好效用具有獨立性質時，線性相加可滿足使用者，也就是說，相加模式必須在先滿足偏好獨立性下才能適用。
- 在許多個案研究中，相加線性模式估計時較乘法線性模式更為有效，而且與實證後的結果很相近，且相加模式較簡易清楚。

4.權重的決定：

模式由不同的權重決定，可分為組合模式及分解模式。

- 組合模式：各方案屬性的權重和偏好值，都是由決策者在訪談時之各問題答案求出，之後利用模式加以組合以求出決策者對方案之偏好。
- 分解模式：只有偏好值是直接由面談或問卷中求出，併同時再問出對方案的偏好值，然後以數學方法加以計算以求得權重，最後利用求出的權重並預測決策者的偏好值。

求取權重的方法可分為二種，一種是配合組合模式，用直接詢問求得，另一種則是配合分解模式，用數學方法計算而得。權重取得的方法，現今已發展出多種方法如消費者評等、點數分配法、線性規劃法。

多屬性效用理論是一個決策分析工具，可以幫助買方的購買決策以買到最滿意最合適的產品，也可以幫助賣方彰顯產品的附加價值，達到差異其產品和服務。因此在衡量協商過程中的提案時，多屬性效用理論很明顯是個有用的工具。因此在本研究中，使用MAUT以衡量協商提案的效用，依據提案效用值加以評估、比較和排序候選方案。

2.6 軟體代理人

2.6.1 代理人定義

以使用者的觀點來看，軟體代理人是一種程式，讓人類委派工作給代理人執行的方式來進行運作，以提供支援(Lange and Oshima, 1998) 。代理人有許多不同類型，以不同方式運作，可以使用在電腦作業系統、資料庫和網路等。然而這些代理人都有一些共同的本質，以系統的觀點來看，一個代理人是一個處於一個執行環境中的軟體元件，並包含以下特點：

- 主要的必備特點有：
  1. 反應性(Reactive)：能感應環境的改變，並作出反應。
  2. 自主性(Autonomous)：能控制自己的行動。
  3. 目標導向(Goal-driven)：是 proactive。
  4. 暫時性持續(Temporally continuous)：持續執行。
- 可能也有以下的特性：
  1. 溝通(Communicative)：可以和其他代理人溝通
  2. 行動(Mobile)：可由一個主機一到另一個主機
  3. 學習(Learning)：能以過去的經驗調適
  4. 可信(Believable)：可為使用者所相信

2.6.2 行動代理人

代理人依其行動特性有無可以分為不移動的代理人和行動代理人。不移動的代理人(Stationary Agent)只能在啟動它的系統環境上執行。假如它需要的資訊不在本身的系統環境上，或它需要跟其他代理人互動時，則它必須透過 Remote Procedure Calling 或 messaging 的溝通機制。而行動代理人(Mobile Agent)不侷限於只能在啟動他的系統環境上執行。它可以在網路的主機(host)間自由旅行。在一個執行環境上被創造出來後，可以把本身的狀態(state 或 attribute values)讓代理人決定該做什麼)和程式碼(code)傳輸到另一個執行環境上繼續執行。

行動代理人的目的就是要提供一個分散處理的架構，讓應用系統的元件可以在不同執行環境之間移動。而這種在網路間動態執行應用元件的能力，便提供應用程式強大的彈性和客制化的能力。Lange and Oshima (1998) 認為使用行動代理人相對於傳統主從架構有七個好處：

1. 降低網路流量負荷(They reduce the network load.)

當大量資料儲存在遠端主機上時，則資料必須在本端被執行，而非傳輸過來執行；也就是將計算移往資料，而非將資料移往計算(Move the computation to the data rater than the data to the computation)。如此便能降低網路上的資料流量。

2. 克服網路延遲問題(They overcome network latency.)

網路延遲對及時系統(Real-time systems)來說是無法允許的。而行動代理人可以克服網路延遲的問題，因為行動代理人可以派遣至 Local 端直接執行。

3. 將協定封裝起來(They encapsulate protocols.)

分散式系統的主機是需要對外送的資料加上協定用的資料，而在接收資料時則將包含的協定資料解譯。當協定改進時，若某端的機器無法迅速更新，很容易會造成延遲現象。行動代理人可以跑到遠端主機，依據其所使用的協定建立所謂的「通道」(Channel)。

4. 非同步自動執行(They execute asynchronously and autonomously.)

行動代理人可以非同步化、自主性地執行，當行動代理人被派遣到遠端後，主機甚至可以關掉或離線，等行動代理人處理完畢之後再連線召回即可。

5. 動態調適的能力(They adapt dynamically.)

行動代理人可以對其執行環境感應並作出自主反應。

6. 容許異質性(They are naturally heterogeneous.)

網路的環境無論是軟體或硬體間都差異很大。而行動代理人是與傳輸無關(transport-layer-independent)也是和硬體無關(computer-layer-independent)，因此提供系統整合的良好能力。

7. 健全與容錯(They are robust and fault-tolerant.)

當一台主機將要關閉時，所有在該主機上的行動代理人將被警告，而且給予時間使它們再派遣到別的主機上繼續執行工作。

2.6.3 尋找代理人

尋找代理人的方法有三種可能性：(Lange and Oshima, 1998)

(1).搜尋(Search)：

搜尋又有幾種方式，一種是送出一個搜尋的代理人(Search Agent)，去拜訪網路上的每個使用端(host)找尋想要找到的代理人；另一種方式是發出詢問多重播送(Multicasting)到各個代理人，然後等候代理人回應。然而，直覺上我們可以發現利用搜尋的方式要在廣大的網路上找到 Agent，看起來不是一個很有效率的方法。

(2).日誌(Logging)：

當一個代理人離開一個使用端(host)，它會留下一個電子指紋，這個指紋會說明他要去哪一個地方，使用端(host)會根據 Aglet 的要求派遣到它想去的地方。然而，日誌有可能被修改而作假，何況維護每個使用端的日誌勢必要耗費不少資源。

(3).註冊(Registration)：

在資料庫註冊代理人的所在位置，而資料庫記載的為最新的資訊，即 Aglet 所在的最新位置。然而使用這個方式會多增加 Aglet 派遣過程的一個程序。